NEWSLETTER
Vol. 29 – No. 2
March - April 2017

Digital Consultants
Rick Sinclair
Curtis Jeung

Current Central Station 3 Version – 1.2.0 (4)
Current Central Station 2 Version – 4.1.2 (3)
Current Mobile Station 2 Version – 2.5

How time flies by! We attended the Rocky Mountain Train Show in early March. Now we are getting ready for the Garden Railway Convention and EuroWest in July.

As we continue our current series, our first article tackles a standard digital upgrade with sound. Our second article begins a new series on line mergers.

Locomotive Upgrades – Sound

In the last issue, I installed a simple digital upgrade. The next step is to do one that is a little more challenging. This time I have chosen an older Austrian Krokodile (#3159).

How I Acquired this Locomotive


Fig. 1 - Märklin 3159 Austrian Krokodile

First, a little background on this loco. I've been interested in this Austrian Krokodile for a long time, and I finally had the chance to buy it (Fig. 1). I had originally made a deal for one that was already digitally upgraded. Well, at the last minute the seller wanted to keep the digital loco and sold me another one he had that was analog.

Since he is a good friend, I didn't mind. I knew I could convert it easily, but naturally I had to make it better than his now. So, I decided to convert it to digital with sound.

Teardown

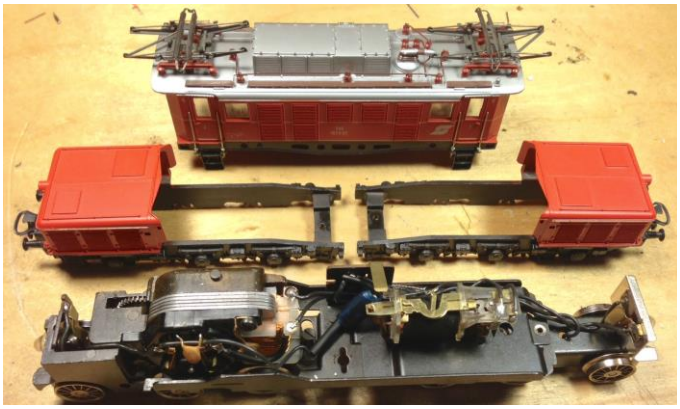

Fig. 2 - Locomotive body is removed

The teardown on this locomotive is as simple as can be. I removed the body to expose the chassis (Fig. 2). After that, I removed all the analog components inside, including the slider and the slider contact plate. I will solder the red decoder wire directly to the slider contact plate. I do this because it is one less point of failure than a splice, and it also looks more professional (Fig. 3).

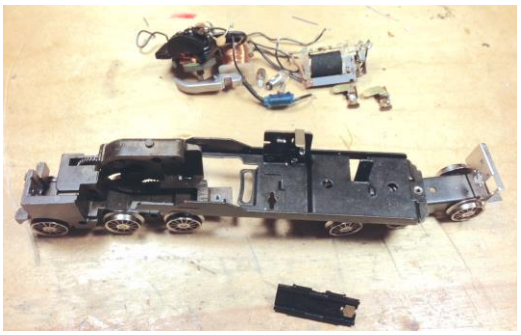

Fig. 3 - Analog components removed

Upgrade Components

For this locomotive, I will be using an mSD/3 (part #60977). This is an MFX decoder that comes programmed with default electric locomotive sounds. Also, I will be using a High-Efficiency Motor Conversion Set (part #60944) for a large flat commutator motor. This motor set comes with two armatures and brush-plates. I simply chose the armature that had the same number of teeth (8 teeth) on the gear and the appropriate brush-plate (Fig. 4).

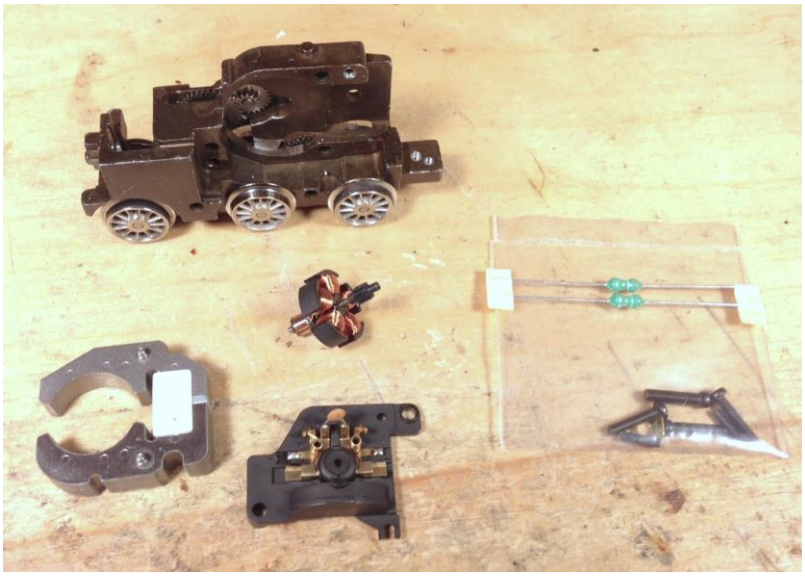

Fig. 4 - Motor components from 60944 set

It is a simple matter to assemble the motor parts into the motor chassis. Remember to install the motor grounding tab under the brush-plate screw. This is also a good time to install the motor brushes (I forgot to install them again) (Fig. 5).

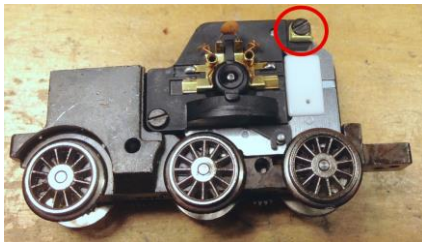

Fig. 5 - Motor ground tab

Decoder Install

Once the motor is complete it gets installed into the chassis and I can work on the “rough-in” of the decoder. This decoder is a 21-pin, so that means it has an adaptor plate and mounting bracket. It also comes with two speakers to choose from (Fig. 6).

Fig. 6 - Märklin mSD/3 components

The mounting bracket attaches to the post that the reverse unit was screwed to, then the adaptor plate snaps in (Fig. 7).

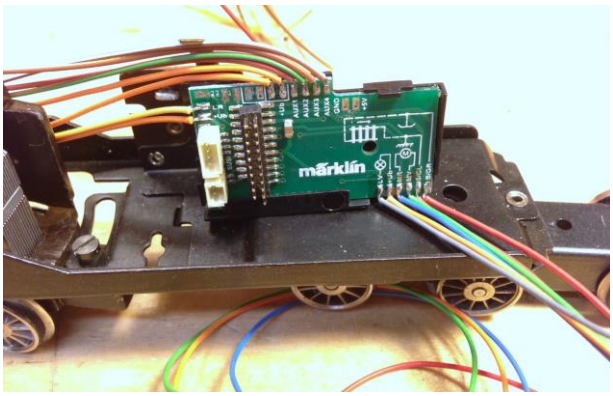
Fig. 7 - Adaptor plate installed

Märklin has done a great job with the adaptor plate. Not only does the decoder have lights and four functions, but the adaptor plate has four return wires (orange wires) for the additional functions or lights. In the past, there was only one return wire and it had to be spliced in, or more wires had to be soldered onto the decoder. These extra return wires are very convenient.

Now that the adaptor plate is installed, I can rough-in the wiring. I used the same light sockets as last time (part #E604180) and tapered the base (Fig. 8), and then


Fig. 8 - Light socket #604180

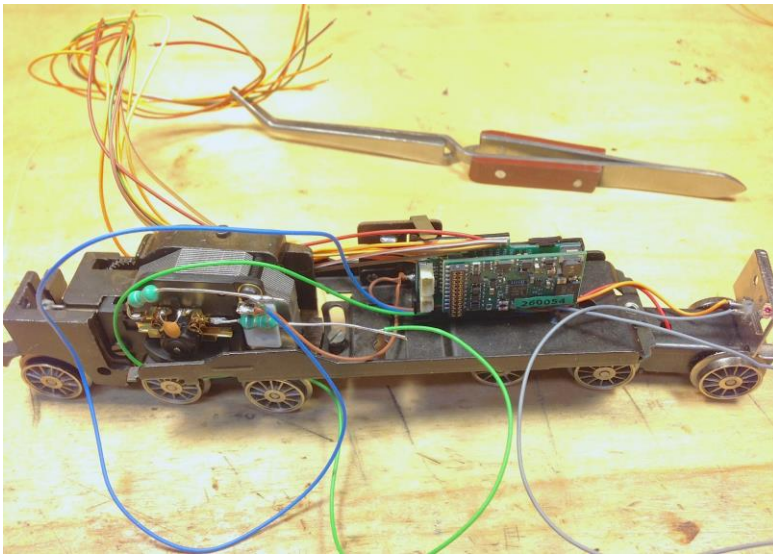
Fig. 9 - Wires roughed-in and decoder installed

pressed them into the light socket holes. Then, I cut the wires to length for the front lights and soldered them to the light sockets. I also soldered an additional chassis ground to the frame. The reason for this is the

motor truck has six wheels and four of them have tires. The two wheels that don't have tires wouldn't be a sufficient ground. So I soldered a wire to the frame to pick up a good ground with the other six wheels on the other truck.

I need to see if the motor turns the correct direction and that the correct light turns on. Since I know that the grey wire is for the forward light, and the #1 engineer's cab will be oriented at that end of the locomotive, I was confident that I could solder the light wires to the socket. The motor wires are "temporarily" soldered in place and the rest are secured so I will not damage the decoder when I test the direction (Fig. 9).

If you remember from the last article, I soldered the "blue" wire to the forward most terminal on the brush-plate and the motor ran backwards. This time, I soldered the "green" wire to the forward terminal. Once I replaced the forgotten motor brushes, the motor ran backwards!

The correct light worked, so all I needed to do was to reverse the motor wires and finish up the rear light connection.

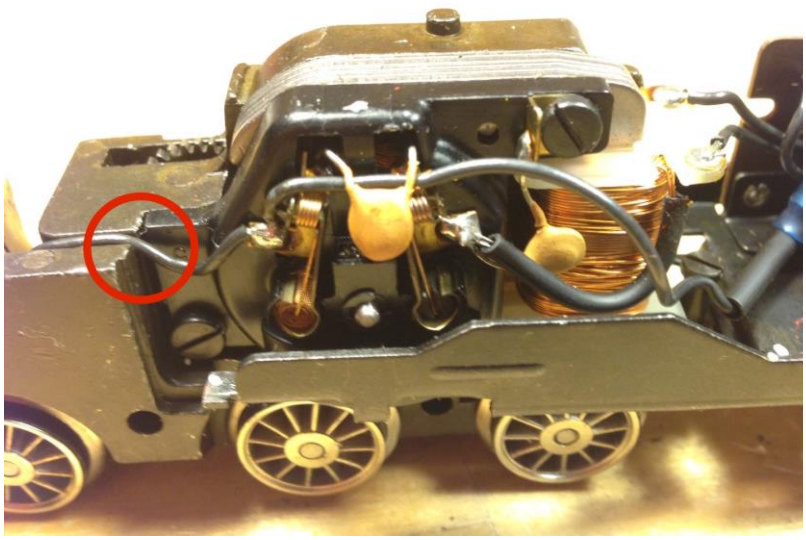

Fig. 10 - Original motor and rear light wire routing

Minor Modification

From time to time, there is a need to do some modifications to the chassis or motor assembly to make them work. I saw the need to do a very minor modification to the brush-plate to get a neat clean look and also to have the factory clearance for the rear wires. In Fig. 10, you can see there is a groove for the rear light wire. Also, the brush-plate allows for the wires to pass in front of it.

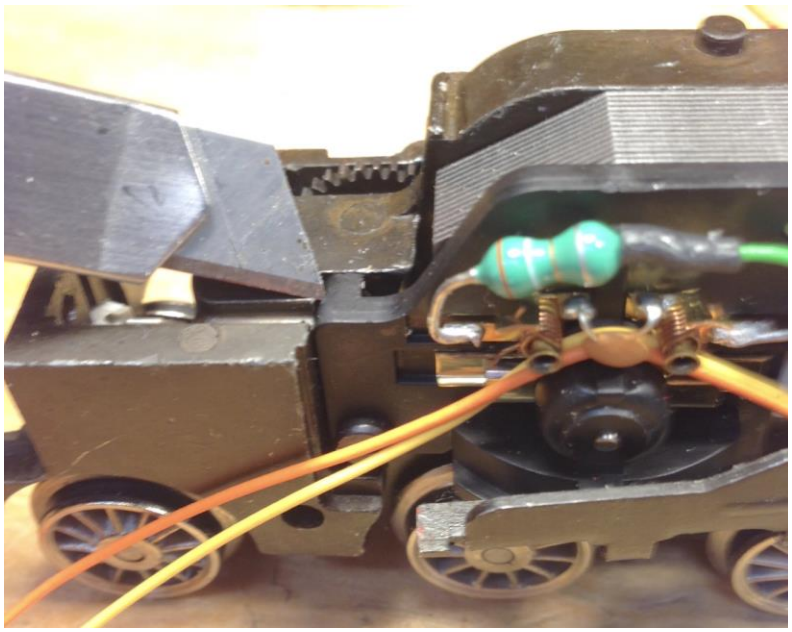

Fig. 11 - Clearance issue for rear light wires

Since worn wire insulation could result in decoder damage, I made a slight modification to the brush-plate. The solution was simple: I used my hobby knife to notch out a corner of the brush-plate (Fig. 11) and ran the wires in the notch (Fig.12).

This solution worked out well. I covered the wires in the groove with some electrical tape to make sure they didn't come out and wear the insulation.

With the decoder tested and the wire clearance solved, I was able to cut the wires to length and solder them.

Since this decoder has so many auxiliary functions that can't be used with this locomotive, I had to remove the auxiliary wires because I couldn't use them (Fig. 13).

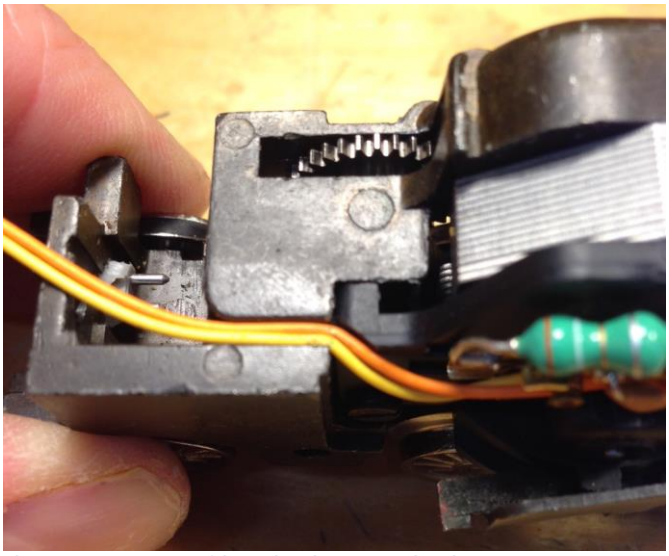

Fig. 12 - Corner of brush-plate notched

Speaker Placement

I had been searching for an appropriate place to mount the large speaker inside the locomotive. I couldn't get it to fit comfortably, so I was forced to use the smaller speaker.

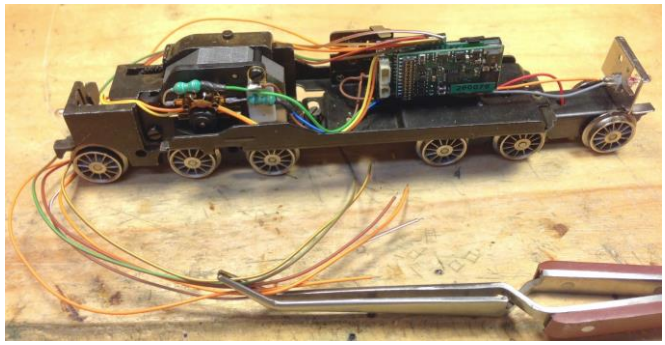

Fig. 13 - Auxiliary wires need to be removed

I mounted the smaller speaker right behind the decoder. There is a strip of double-sided tape to mount the speaker. Once mounted, it's just a matter of plugging it into the decoder (Fig. 14).

I routed the wires so that they would not get pinched and then they are tied (Fig. 15).

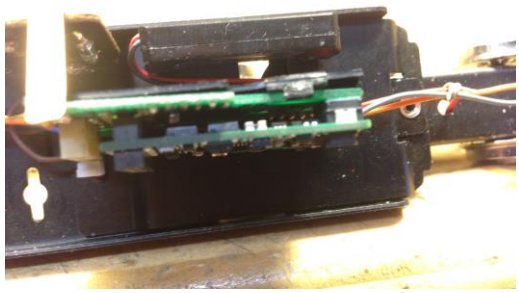

Fig. 14 - Speaker placement

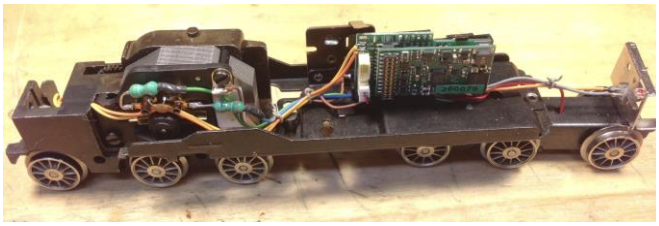

Fig. 15 - Wires routed and tied

Decoder Parameters

With the Central Station I can set the driving characteristics, sound volumes and give it a new name. I was thinking of giving it the name “Better than yours (friend name here)” – you know who you are!

Once I have made these changes, I will write them to the decoder.

Now that this locomotive is better than the one my friend has, I will have to take it over to his house and run it. The finished loco runs well and the sounds are great (Fig. 16). I am very impressed with the small speaker and as always the 5-pole motor runs smooth.

Fig. 16 - Finished locomotive

Parts Used

The list of parts that I used for this conversion is:
1 - 60977 mSD/3 Sound Decoder
1 - 60944 5-Pole High-Efficiency Motor
2 - E604180 Light Sockets
2 - E610080 Light Bulbs

Märklin uses their standard wire colors on the adaptor plate:

Red – Slider
Brown – Chassis Ground
Orange – Auxiliary/Light Return
Green – Motor
Blue - Motor
Grey – Forward Light
Yellow – Reverse Light

The auxiliary function wires are colored as follows:

Orange – Auxiliary/Light Return
Brown/Red – Auxiliary 1
Brown/Green – Auxiliary 2
Brown/Yellow – Auxiliary 3
Brown/White – Auxiliary 4

Enjoy your hobbies!

Rick Sinclair

Line Mergers, Part 1 – Branch Line

Probably the most problematic sort of automation with memory scripts involves the merging of branch lines or exchanges onto a single track line. Variables like priority, direction of travel and release points all need to be considered in order to avoid collisions. In this article, I hope to address some of the common situations and possibly highlight factors that you can look for and adjust for your particular situation.

Branch Line Entry

Usually a branch line isn't an automated task, but it can be a 'set and forget' setting where you'll only need to start the train off a branch line. It is also a good building block to examine the control sections that will regulate the interactions of the two joining lines.

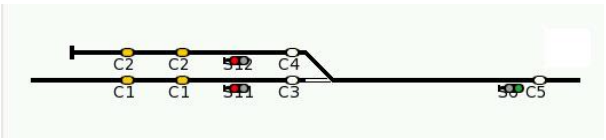

Fig. 1 - Branch line layout with contact sensor and signal placement

Fig. 1 illustrates the necessary contact tracks ('C*x*') for a branch line entry control. The contacts labelled C1 & C2 are each shown on the layout in a dual place setting, but they are each a single contact area. They are doubled to show, for this tutorial, that they will be used as 'occupational' signals. In other words, they are to signify whether or not a train is waiting at a signal stop.

Contacts C3 and C4 will be consigned as entry (or arrival) triggers. Their purpose is to signal when a train has begun to enter the merging turnout. It will set the opposing signal to 'stop' aspect, thus preventing an alternate-line train from entering into the merging turnout. See Table 1.

Name	C3	C4
Script steps	Signal on branch 'stop'	Signal on main 'stop'

Table 1 – Line scripts prior to merge

Contact C5 is the 'block clear' trigger. It is used as the 'okay' signal to show that the block between the turnout and its adjacent signal is empty and ready to receive either a train from the branch or the main line. It is configured as an exit (or departure) type trigger. The 'block clear' (bc) trigger when used in only a main line operation is our usual release script, which allows a trailing train to enter its newly vacated block. However, in a branch line situation C5 must now determine if a train is in wait on contacts C2.

At this point, I should mention that in merging situations you must decide on track priorities. If for instance, you had a train waiting at both tracks (C1 and C2), you would have to decide which train has priority to enter over the other. For the branch line on a model railroad, I would give priority to the branch line because is not a common interruption in the main line operation, and I would have manually set the train in motion to enter the main line to begin with. I mention track priorities now with this simple setup, because this is the area which will become more complex and problematic in the subsequent examples.

The mental logic which will be used on the C5 trigger is, "Is there a train on contact C2? If so, then I will set the signal on the branch line to release that train, while keeping the signal on the mainline at 'stop'."

By asking the question above, I am creating a condition which I need to determine, before I can take action. Until now, I have only touched lightly on how we can use the Central Station 2 to create conditions to determine script actions. These conditions can be written into your memory scripts by accessing the 'Advanced settings for Route __' window. This is found by clicking on the 'Ext.' icon at the bottom of the keyboard configuration page (Fig. 2).

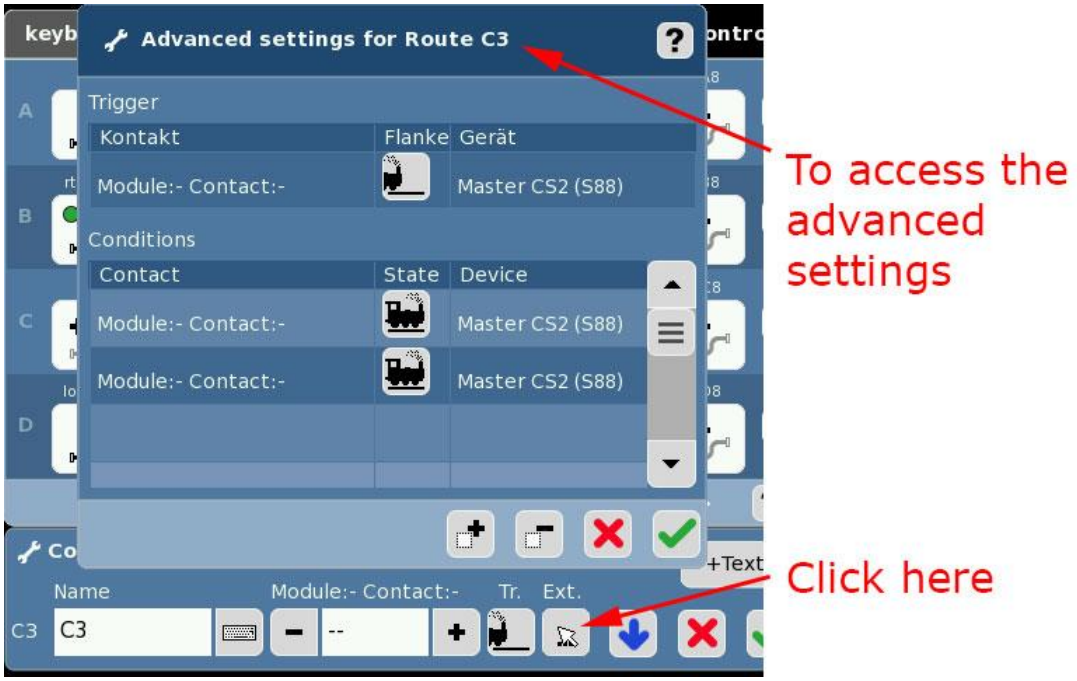

Fig. 2 - Accessing advanced settings for scripts

You can see the window is subdivided in two sections, the Trigger section (top) and the Conditions section (bottom). The trigger settings are identical to that of the settings next to the 'Ext' button (see very bottom of Fig. 2). It is now recommended that you always set the trigger settings in this window instead of the Configuration window, because the 'Gerat' or Device settings have changed when using newer L88/S88 with the CS2.

The Conditions settings are where you will configure the ability to use track occupations to evaluate the execution of the assigned memory script. In our branch line example: I would set the first line under the Conditions>Contact table to the contact ID: '2' (C2 from Fig. 1). Under the 'State' column click on the icon to set it for occupied status (Locomotive on track). The conditions set in this window must first be true in order for the memory script assigned to this trigger to execute. This is how the CS2 mimics the mental logic that I wrote about earlier.

Fig. 3 - Conditions setting showing track occupation reading for Contact 2

The equivalent notes that I may write to illustrate this logic can be found in Table 2. I can see under the name, that the script is for a Contact 5 departure, with an 'Ext' condition set to evaluate Contact 2. I can tell at a glance where to troubleshoot for sensor errors. This shorthand is also necessary for the CS2 with the limited space available for titling in the Memory pages.

Name	C 5d X 2
Steps	Signal on main to 'stop'
	Signal on branch to 'go'

Table 2 – Conditional script notation

The logic applied in Table 2 is a basic start to prevent collisions. However, the operational example still needs some additional consideration before creating a faultless automation. Table 3 includes a basic single line block script 'C 5d' and our first conditional script 'C 5d' from Table 2.

Name	C 5d	C 5d X 2
Steps	Signal on main to 'go'	Signal on main to 'stop'
		Signal on branch to 'go'

Table 3 – Two memory scripts assigned to Contact 5

Note that we now have two separate scripts for Contact 5 departure trigger ('C 5d'). Also, note that both scripts have opposing instructions for the signal on the main (line). One sets the signal to 'go', the other script sets the signal to 'stop'. I have highlighted this in red. Running two simultaneous scripts from the same trigger is fine as long as you have conditions set to avoid these conflicts. Remember that 'C 5d X 2' will only operate if a train is occupying the sensors at C2 (see Fig 1). Therefore, the script 'C 5d X 2' will only operate in special conditions.

The script 'C 5d', will operate at every instance because there is no regulating condition. Unfortunately, this will not be appropriate if the script for 'C 5d X2' needs to activate, because the combination of both scripts allows for both the branch and main line trains to proceed. Therefore, we must make alterations to the script for 'C 5d'. Fig. 4 Illustrates the modifications to the script on 'C 5d'.

Fig. 4 - C 5d conditions reading unoccupied branch contact at C2

I have made a naming change to both my scripts to tell me the differences between the two. Table 3 (in red). 'C 5D' is now 'C 5d X 2u' to indicate that I now have a condition ('X') based on an unoccupied section of track at contact sensor C2 ('2u'). In Fig. 4, notice how the Conditions now look for unoccupied sensors at contact 2, unlike the same for 'C 5d X 2o' (compare with Fig. 3).

Name	C 5d X 2u	C 5d X 2o
Steps	Signal on main to ‘go’	Signal on main to ‘stop’
	Signal on branch to ‘go’	Signal on branch to ‘go’

Table 4 – Clarifications to script IDs and default setting of branch signal.

In Table 4, I have also added the command (in red) to set the signal on the branch line to go. This was necessary as the default to set both signals to ‘go’, otherwise our branch signal will get stuck as a ‘stop’ signal when all lines are clear. *(Note: In a standard block script, only the main line signal would be set, but we now have two signals)*. It is important to realize that scripts with conditions are completely ignored when conditions are not met. Therefore, while the two scripts have identical steps, only one script will have executed (due to the occupied/unoccupied status of sensor C2).

In Table 5, I have summarized all the scripts needed to operate a branch line automation. You may have noticed that there is no script that calls for the sensors at C1. They are not required for a branch line setup, and it is not necessary to have included them in the layout example.

Name	C 3a	C 4a	C 5d X 2u	C 5d X 2o
Script Settings	Sig Branch ‘stop’	Sig Main ‘stop’	Sig Main ‘go’	Sig Main ‘stop’
			Sig Branch ‘go’	Sig Branch ‘go’

Table 5 – Complete list of script step settings required for a branch line.

Setting up conditions for merging lines expands in complexity based on purpose. While I wish to dig deeper into various applications, I feel this article may call for you to experience, experiment and play with the concepts written. It only scratches the surface of how to handle two trains onto a single line. The branch line example only introduces the issues, and with it, the most simple of resolutions. In my next automation article, I will try to address when two main lines try to converge. Thanks for reading, and don’t hesitate to contact me if you have any questions.

Curtis Jeung

Upcoming appearances:

National Garden Railway Convention Train Show
Renaissance Tulsa Hotel & Convention Center
6808 S 107th East Ave Tulsa, OK
July 15, 2017

EuroWest
Hiller Aviation Museum
601 Skyway Rd San Carlos, CA
July 22-23, 2017

Trainfest
Wisconsin State Fair Park Expo Ctr
West Allis (Milwaukee), WI
November 11-12, 2017

To contact Curtis and Rick for help with your Digital, technical and product related questions:

Phone: 650-569-1318 Hours: 6:00am – 9:00pm PT, Monday through Friday.

E-mail: digital@marklin.com

Get Connected with Märklin Digital on Facebook at this link:
https://www.facebook.com/Marklin-Digital-201480640231441/?fref=ts